Progressive Knowledge Graph Completion

Anonymous ACL submission

Abstract

Knowledge Graph Completion (KGC) has emerged as a promising solution to address the issue of incompleteness within Knowledge Graphs (KGs). Traditional KGC research primarily centers on triple classification and link prediction. Nevertheless, we contend that these tasks do not align well with real-world scenarios and merely serve as surrogate benchmarks. In this paper, we investigate three crucial processes relevant to real-world construction scenarios: (a) the *verification process*, which arises from the necessity and limitations of human verifiers; (b) the *mining process*, which identifies the most promising candidates for verification; and (c) the *training process*, which harnesses verified data for subsequent utilization; in order to achieve a transition toward more realistic challenges. By integrating these three processes, we introduce the Progressive Knowledge Graph Completion (PKGC) task, which simulates the gradual completion of KGs in real-world scenarios. Furthermore, to expedite PKGC processing, we propose two acceleration modules: Optimized Top-k algorithm and Semantic Validity Filter. These modules significantly enhance the efficiency of the mining procedure. Our experiments demonstrate that performance in link prediction does not accurately reflect performance in PKGC. A more in-depth analysis reveals the key factors influencing the results and provides potential directions for future research.

1 Introduction

Knowledge Graphs (KGs) have wide-ranging applications across diverse domains, including question answering (Mohammed et al., 2018), information extraction (Han et al., 2018), and recommender systems (Zhang et al., 2016). Nevertheless, KGs frequently grapple with incompleteness, resulting in the absence of critical factual links (Galárraga et al., 2017). Consequently, Knowledge Graph Completion (KGC) assumes a pivotal role in automating the enhancement of KGs (Sun et al., 2019b).

In the past, tasks such as link prediction and triple classification required predicting the tail entity of a query and judging the correctness of the proposed triples, respectively. However, where do these effective queries come from in real-world scenarios? No previous work has explored this issue. Furthermore, the performance of models in these tasks falls short of the high accuracy requirements of KG, as confirmed by prominent companies like Google (Pan et al., 2023). Therefore, we advocate for simulating a more realistic scenario in KGC. To meet the requirements for stringent knowledge precision, it is necessary to incorporate a *verification process* to simulate human-machine collaboration. Additionally, human verifiers face inherent limitations in their daily data processing capacity, underscoring the need for a *mining process*. Within this process, KGC models curate a specific quota of the most promising facts. These freshly acquired facts, in turn, facilitate the iterative refinement of KGC models through a *training process*. Finally, we iteratively implement these three processes to form a progressive completion.

Drawing from the above reasons, we introduce the Progressive Knowledge Graph Completion (PKGC) task. PKGC emulates the gradual process of KG completion, as depicted in Figure 1. It commences by training a model using a KG and subsequently invokes the model to identify the most promising facts. Thereafter, a limited verifier selects the true facts, integrating them into the KG.Furthermore, we have made substantial strides in expediting the mining process, particularly vital given the exponential growth in potential facts as KGs expand in size. To address this, we introduce two modules: Optimized Top-k and Semantic Validity Filter (SVF), both of which bring about a significant reduction in both space and time complexity. The former implements root filtering within a heap structure and incorporates a batch warm-up strategy, while the latter capitalizes on

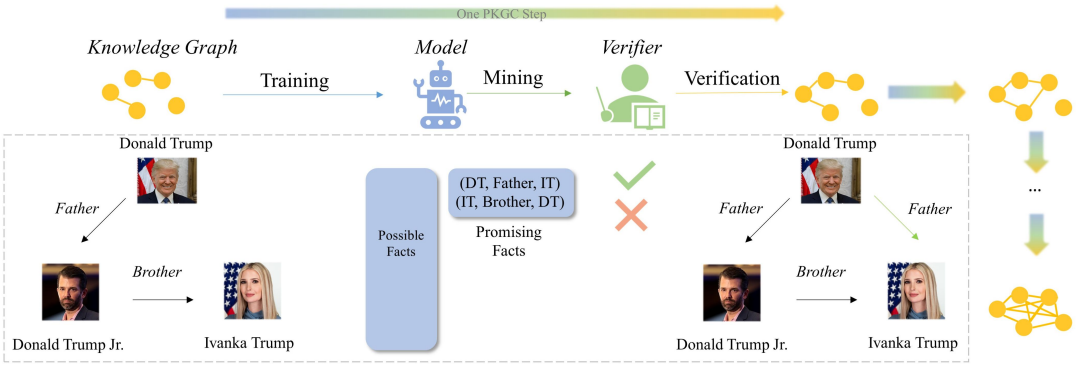


Figure 1: PKGC consists of training, mining and verification procedures. The knowledge proposed by the KGE model will be added to the knowledge base after verification.

the semantic properties inherent to KGs.

In the realm of PKGC, we propose two novel metrics specifically tailored to provide a more realistic evaluation of the model's performance. It becomes apparent that relying on metrics derived from previous Surrogate Knowledge Graph Completion (SKGC) tasks inadequately guides the model's performance in PKGC. Therefore, they are not suitable for selecting models in real scenarios. In light of this observation, we initiate a discussion to delve into the underlying factors influencing model performance in PKGC. We also explore opportunities to leverage verified knowledge in PKGC, as opposed to SKGC, which models entities and relations in a one-off manner. We leverage two simple incremental learning approaches and throw light on them for future research.

2 Related Work

Knowledge Graph Embedding KGC encompasses a diverse array of methods (Ji et al., 2021), including embedding-based (Bordes et al., 2013; Wang et al., 2014b; Sun et al., 2019a; Zhang et al., 2022), rule-based (Galárraga et al., 2013; Omran et al., 2021), and reinforcement learning-based (Xiong et al., 2017; Lin et al., 2018). Given the prevalence of the embedding-based approach, often referred to as Knowledge Graph Embedding (KGE), this article primarily centers its focus on this methodology. Notably, KGE methods have consistently dominated the leaderboards of competitions like ogb-wiki2 (Hu et al., 2020), a prominent fixture in the field of KGC. To streamline the scope, this paper confines its inquiry to the realm of structural information, excluding description-based approaches (Xie et al., 2016; Wang et al., 2014a; An et al., 2018; Wang et al., 2021; Yao et al., 2019) within KGE. The pioneering distance-based model in KGE, TransE (Bordes et al., 2013), conceptualizes each relation as a translation operation. TransR (Lin et al., 2015) and TransH (Wang et al., 2014b) employ projections to handle complex relations, while MuRP (Balazevic et al., 2019) extends the modeling space to hyperbolic geometry to capture hierarchical structures within KGs. In addition to translation, RotatE (Sun et al., 2019a) introduces rotation as a means to represent relations. RotH (Chami et al., 2020) further amalgamates these two operations into hyperbolic space. RESCAL (Nickel et al., 2011) stands as the inaugural bilinear-based model, while CP (Hitchcock, 1927) simplifies relation matrices to diagonal representations. Deep learning-based models harness convolutional neural networks (Dettmers et al., 2018; Nguyen et al., 2018), Transformers (Chen et al., 2021; Vaswani et al., 2017), and graph neural networks (Schlichtkrull et al., 2018; Wang et al., 2020; Liu et al., 2021; Tan et al., 2023) as encoders, synergizing them with the aforementioned decoder models.

3 Methodology

This section initiates with a comprehensive exposition of the progressive completion task, which is elaborated upon in Section 3.1. Following this, in Section 3.2, we introduce two acceleration techniques.

3.1 Progressive Knowledge Graph Completion

To commence, PKGC initiates by partitioning a KG, denoted as $\mathcal{F}$, into two components: the known portion, refered to as $\mathcal{F}_{known}$, and the unknown portion, labeled $\mathcal{F}_{un}$, based on a predefined

ratio ρ. This procedure is in alignment with SKGC practices. Distinguishing itself from SKGC, PKGC systematically advances by expanding the known segment, $\mathcal{F}_{known}$, through a sequence of incremental verifications executed by a verifier denoted as ψ, with the capability to authenticate n_c candidate facts during each iteration. This process adheres to a cyclic routine that encompasses training, mining, and verification procedures, as elaborated in Algorithm 1. (Detailed comparison is in Appendix A)

Algorithm 1 Progressive Knowledge Graph Completion

Input: KG $\mathcal{F}$, ratio ρ, verifier ψ, maximum step n_s
Output: Updated KG $\mathcal{F}_{known}$, metrics $\mathcal{M}$

Initialization: $s(\cdot) \leftarrow PretrainModel$, $\mathcal{F}_{known}$,$\mathcal{F}_{un} \leftarrow SplitKG(\rho)$, $i \leftarrow 0$, $\mathcal{F}_{visited} \leftarrow \mathcal{F}_{known}$
for $i = 1, 2, \ldots, n_s$ **do**
  if update condition **then**
    $s(\cdot) \leftarrow UpdateModel$
  end if
  $\mathcal{F}_c \leftarrow KnowledgeMining$
  $\mathcal{F}_{new} \leftarrow Verification$
  $\mathcal{F}_{visited} \leftarrow \mathcal{F}_{visited} \cup \mathcal{F}_c$
  $\mathcal{F}_{known} \leftarrow \mathcal{F}_{known} \cup \mathcal{F}_{new}$
end for
$M \leftarrow CalculateMetrics$

Within the training phase, we train a model denoted as $s(\cdot)$ using $\mathcal{F}_{known}$, mirroring the methodology employed in Surrogate Knowledge Graph Completion (SKGC). Given that the Knowledge Graph undergoes alterations following each step, $s(\cdot)$ can be subject to a range of update techniques to adapt to these modifications.

The mining phase entails $s(\cdot)$ generating a set of n_c candidate facts, $\mathcal{F}_c$, derived from the entire spectrum of conceivable facts, excluding those present in $\mathcal{F}_{known}$ or validated within $\mathcal{F}_{visited}$.

In the verification phase, the verifier ψ systematically scrutinizes the authenticity of the candidate facts within $\mathcal{F}_c$ and subsequently incorporates the verified facts into the established KG, $\mathcal{F}_{known}$. In the context of our experiments, this process is emulated by verifying the existence of facts within $\mathcal{F}_{un}$.

To assess the performance of PKGC, we establish a predefined maximum number of steps, denoted as n_s, and subsequently evaluate the metrics detailed in Section 4.1 upon the completion of these steps.

3.2 Acceleration for Mining Process

Given that the mining process can be fundamentally likened to a top-k procedure, executing it without optimization proves to be exceedingly impractical due to the sheer volume of potential facts involved.

To illustrate the magnitude of this issue, consider a KG housing $|\mathcal{E}|$ entities and $|\mathcal{R}|$ relations. Such a KG spawns an astronomical $|\mathcal{E}|^2|\mathcal{R}|$ potential facts, rendering the storage and effective sorting of these facts unviable through conventional means. In response to this formidable challenge, we introduce two purpose-built modules: Optimized Top-k and SVF. These modules are meticulously crafted to achieve substantial reductions in the time and space resources necessary for mining while preserving efficiency and effectiveness.

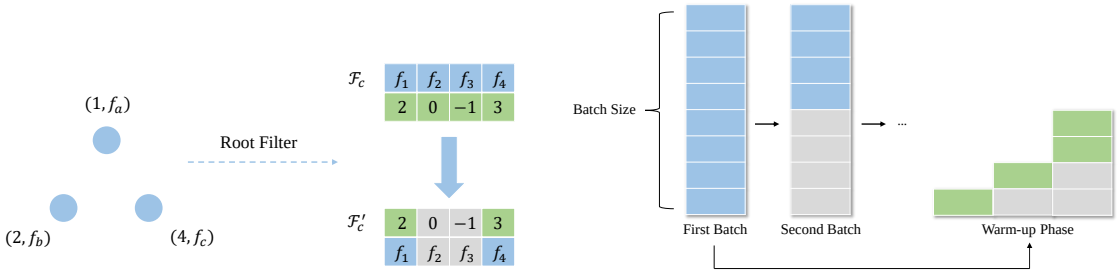
3.2.1 Optimized Top-k algorithm

In order to obviate the necessity of retaining scores for every conceivable triple, we have elected to implement a heap structure, which entails only a modest additional space of $O(k)$. Nevertheless, the unrefined heap-based top-k algorithm endeavors to establish a min-heap and proceeds to insert each potential triple into the heap contingent upon its respective score. Notably, despite the algorithm's time complexity of $O(n\log(k))$ (Cormen et al., 2022), its practicality is hampered by the vast scale of $|\mathcal{E}|^2|\mathcal{R}|$.

To address this issue, we introduce an optimized top-k algorithm that builds upon the fundamental approach. This algorithm comprises two intricately interconnected components: the root filter and the warm-up.

Root Filter In light of the impracticality of acquiring scores for all conceivable facts in a single sweep, we adopt a strategy of segregating them into a sequence of incremental batches. This sequential approach offers us a unique opportunity to refine the filtration process within each batch by leveraging insights gleaned from previous batches.

As depicted in Figure 2a, we possess the capability to exclude facts with scores inferior to that of the root element within the heap. This filtration procedure can be executed in parallel, resulting in a substantial acceleration in processing speed without overlooking any candidate facts.



(a) Root filter.

(b) Batch warm-up.

Figure 2: Figure (a) depicts the Root filter process, where lower-scoring triplets are directly filtered out. On the right, Figure (b) demonstrates Batch warm-up. In this process, the data from the initial batch undergoes decomposition, and its size gradually increases until it reaches a predetermined limit. And subsequent batches maintain a consistent size.

warm-up While the root filter significantly expedites the processing of subsequent batches, its effectiveness does not extend to the initial batch. We have observed that when dealing with a sizable batch size, the execution of the first batch experiences notable delays. To address this challenge and to maximize the utility of the root filter, we have introduced a warm-up schedule for the batch size within each mining process. Specifically, we initiate with a batch size of 1 and systematically increase it exponentially until it aligns with the intended batch size, as illustrated in Figure 2b. This approach is designed to mitigate the performance issues associated with larger batch sizes during the initial stages of the mining process.

3.2.2 Semantic Validity Filter

In addition to delving into the mining process, we tackle the aspect of "what to mine" by introducing a pivotal component, the Semantic Validity Filter (SVF), which efficiently trims down the search space. Our inspiration for this derives from the recognition that not all combinations of entities and relations are valid (Li and Yang, 2022). For instance, the head entity of the relation 'isCityOf' can only be the name of a 'city', and not a 'human'.

The SVF implementation entails acquiring class information for each entity and maintaining a set that documents each pairing of class and relation initially present in the initial known KG $\mathcal{F}_{known}$. Throughout the mining process, we exclusively entertain queries that align with the SVF criteria, effectively eliminating invalid queries and significantly reducing the search space. It's worth noting that in cases where certain entities lack class information during data collection, we abstain from imposing constraints on queries for these entities.

Algorithm 2 Optimized Top-k with SVF

Input: Known Facts $\mathcal{F}_{known}$, Visited Facts $\mathcal{F}_{visited}$, model $s(\cdot)$, mining maximum batch size b_m^{max}, class dict $\mathcal{D}$, number of candidate n_c

Output: Mined candidate knowledge $\mathcal{F}_{mined}$

Initialization: Batch begin index $b_b \leftarrow 0$, Batch size $b_m \leftarrow 1$, min-heap H with n_c dummy elements f_d that $s(f_d) \leftarrow -\infty$
$\psi \leftarrow GetSVF(\mathcal{F}_{known}, \mathcal{D})$ *// SVF*
$Q \leftarrow GetValidQueries(\mathcal{F}_{known}, \psi)$
while $b_b \leq Q.size$ **do**
  $q \leftarrow Q[b_b : b_b + b_m]$
  $\mathcal{F}_c \leftarrow GetCandidateFacts(q)$
  $\mathcal{F}'_c \leftarrow \{f | f \in \mathcal{F}_c, s(f) > s(H.root)\}$ *// Root Filter*
  for triple $f \in \mathcal{F}'_c$ **do**
    if $f \notin \mathcal{F}_{visited}$ **and** $s(f) > s(H.root)$ **then**
      $H.replace(f)$
    end if
  end for
  $b_m \leftarrow max(b_m^{max}, 2 * b_m)$ *// warm-up*
  $b_b \leftarrow b_b + b_m$
end while
$\mathcal{F}_{mined} \leftarrow Set(H)$

We present the operational details of these modules within Algorithm 2.

4 Experiment

In this section, our discussion unfolds in a structured manner. We initiate by introducing the experimental setup, elaborated upon in Section 4.1.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{K}_{known}\rvert$	$\lvert\mathcal{K}_{un}\rvert$	ρ
FB15k	14,951	1,345	532,989	59,221	0.9
WN18	40,943	18	106,009	45,433	0.7

Table 1: Statistics of Two Benchmark Datasets.

Subsequently, we unveil the key findings in Section 4.2, culminating in a thorough analysis of the observed phenomena in Section 5.

4.1 Experiment Setting

Datasets Our datasets are meticulously crafted, rooted in FB15k and WN18 (Nickel et al., 2011). To maintain a consistent initial completion ratio, denoted as ρ, we partition the dataset into two distinct categories: the initial triples, $\mathcal{K}_{known}$, and the unexplored triples, $\mathcal{K}_{un}$. During this data partitioning process, we impose stringent constraints to ensure that undetermined entities and relations do not appear in $\mathcal{F}_{un}$. Comprehensive dataset statistics are thoughtfully presented in Table 1. It's noteworthy that we consciously opted against utilizing the FB15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018) datasets, favoring a pursuit of more lifelike scenarios. Our primary research focus pivots not on the dataset's inherent complexity, but rather on the nuanced exploration of basic model performance within the context of real-world scenarios.

Partition In the process of partitioning the dataset, our primary objective is to ensure the comprehensive inclusion of all entities and relations within $\mathcal{K}_{known}$. This goal guides a meticulous procedure, during which we thoroughly scrutinize each triple, diligently recording the entities and relations that make an appearance. In the event that a triple is encountered, containing components that have not yet found a place in our records, it is promptly added to the scaffold set. The remaining triples undergo a division into two distinct segments, this division hinging on the predetermined ratio ρ. One segment becomes $\mathcal{K}_{un}$, while the other segment is integrated into the scaffold set to form $\mathcal{K}_{known}$. For an exhaustive and step-by-step elucidation of this partitioning process, we direct your attention to Algorithm 3.

Baselines Our study encompasses a comprehensive comparative analysis between our task and the well-established traditional structure-based models, which can be categorized into two primary groups: distance-based models and bilinear models. The distance-based models considered in this study are TransE (Bordes et al., 2013), RotatE (Sun et al., 2019a), and RotE (Chami et al., 2020). On the other hand, the bilinear models comprise RESCAL (Nickel et al., 2011), CP (Hitchcock, 1927), ComplEx (Trouillon et al., 2016), QuatE (Zhang et al., 2019), and UniBi (Li et al., 2023). UniBi stands out for its unique approach of normalizing both the modulus of the entity vector and the spectral radius of the relational matrix to 1.

Algorithm 3 Dataset Partition

Input: Total Facts $\mathcal{F}_{total}$, initial ratio ρ
Output: Known Facts $\mathcal{F}_{known}$, Unexplored Facts $\mathcal{F}_{un}$

Initialization: Scaffold triples set $S \leftarrow \phi$, visited entity set $S_e \leftarrow \phi$, visited relation set $S_r \leftarrow \phi$
for $(h, r, t) \in \mathcal{F}_{total}$ **do**
  if $h \notin S_e$ **or** $t \notin S_e$ **or** $r \notin S_r$ **then**
    $S_e.add(h)$
    $S_e.add(t)$
    $S_r.add(r)$
    $S.add((h, r, t))$
  end if
end for
$n \leftarrow len(\mathcal{F}_{total}) * \rho$
$S_{remain} \leftarrow \mathcal{F}_{total} - S$
$\mathcal{F}_{un} \leftarrow S_{remain}[: len(\mathcal{F}_{total}) - n]$
$\mathcal{F}_{known} \leftarrow S_{remain}[len(\mathcal{F}_{total}) - n :] + S$

Training The PKGC training process consists of three phases: (1) Hyperparameter optimization, (2) Model training, and (3) Model update. During the initial phase, we partition the set of known facts, $\mathcal{F}_{known}$, into training and validation subsets to perform hyperparameter optimization, following the approach established in prior SKGC research. In the second phase, the model is trained using the complete dataset of $\mathcal{F}_{known}$. In the final phase, the model is updated based on verified facts. This paper primarily focuses on the first two phases, with the third phase introduced in Section 5.3. In these first two phases, we adopt a reciprocal setting wherein we generate a new reciprocal triple (t, r', h) for every (h, r, t) in $\mathcal{F}_{known}$. To enhance the model's expressiveness, we introduce the DURA and Frobenius norm as regularization terms with a positive weight parameter, denoted as $\lambda > 0$. Descriptions about hyperparameters are in Appendix C.

	WN18				FB15K			
Model	MRR	Hits@10	MOAR	CR@50	MRR	Hits@10	MOAR	CR@200
TransE	0.399	0.590	0.223	0.803	0.487	0.731	0.083	0.913
CP	0.563	0.687	0.515	0.920	0.529	0.749	0.588	0.987
RotatE	0.595	0.699	**0.768**	**0.935**	0.543	0.756	0.621	0.987
RotE	0.596	0.705	0.765	**0.935**	0.540	0.761	0.317	0.945
ComplEx	0.561	0.692	0.630	0.927	0.547	0.765	0.623	0.986
QuatE	0.596	0.704	0.759	0.933	0.547	0.767	0.637	0.984
RESCAL	0.548	0.661	0.545	0.892	0.450	0.684	0.668	0.988
UniBi-O(2)	0.598	0.712	0.761	0.934	**0.550**	**0.771**	0.841	0.994
UniBi-O(3)	**0.599**	**0.716**	0.762	0.934	0.548	0.768	**0.846**	**0.995**

Table 2: Results of different models on both previous and progressive metrics. Best results are **bold**, second ones are underlined.

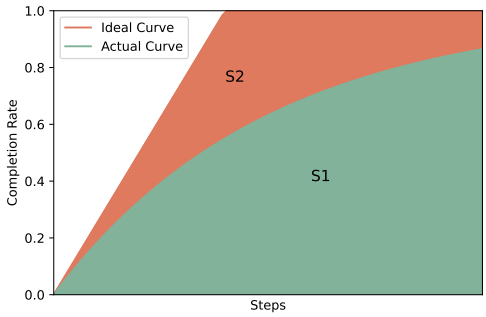


Figure 3: Illustration of how to calculate MOAR, which is the ratio of the area enclosed by the actual curve and the ideal curve, respectively.

$$\mathcal{L} = -\sum_{(h,r,t)\in\mathcal{F}_{train}} \log\left(\frac{\exp(s(h,r,t))}{\sum_{t'\in\mathcal{E}}\exp(s(h,r,t'))}\right) + \lambda \cdot Reg(h,r,t) \quad (1)$$

In Equation 1, we include an additional scalar parameter $\gamma > 0$ after $s(\cdot)$ to inherit the setting from UniBi. We give details of the hyperparameter settings in Section C.

During the mining and verification phases, a scatter chart is constructed to depict the relationship between the completion ratio and the number of steps, guided by $\mathcal{K}_{un}$. We designate the ultimate completion ratio achieved at step k as the critical metric CR@k. Furthermore, we employ the Area Under the Curve (AUC) of the ideal and practical completion curves to quantify an additional indicator, the Model-to-Oracle Area Ratio (MOAR), as illustrated in Figure 3 and computed using Equation 2.

$$MOAR = \frac{S_1}{S_1 + S_2} \quad (2)$$

We fix k at 200 for the FB15k dataset and at 50 for the WN18 dataset, accounting for variations in the performance of the foundational models on these datasets.

4.2 Main Result

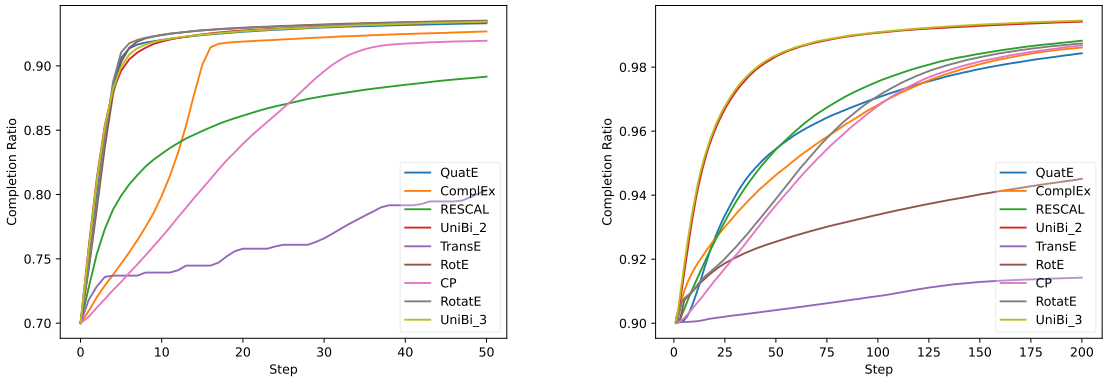
The performance of the models on WN18 and FB15k is presented in Table 2. Additionally, we provide a detailed illustration of the dynamics of the models on these datasets through Figure 4a and Figure 4b to enhance our understanding of the PKGC process.

Initially, we conduct separate analyses of the models' performance on SKGC and PKGC. Our findings indicate that most models exhibit strong performance on both datasets within the realm of SKGC, whereas this extends to only one dataset within PKGC. Notably, UniBi stands out as the sole model demonstrating commendable performance across all metrics and datasets.

Furthermore, we note that the performance of models in link prediction does not consistently align with their performance in PKGC. On one hand, the models' performance in link prediction and PKGC sometimes diverges. For instance, RESCAL exhibits the lowest Mean Reciprocal Rank (MRR) on FB15k but secures the third-best result in Completion Ratio (CR). On the other hand, models that perform similarly in link prediction may exhibit differences in PKGC performance. A case in point is the closeness in MRR between UniBi-O(3) and ComplEx (0.548 vs. 0.547), yet they demonstrate disparities in CR (0.995 vs. 0.986). Notably, QuatE and UniBi are the only models performing well across all metrics in both datasets.

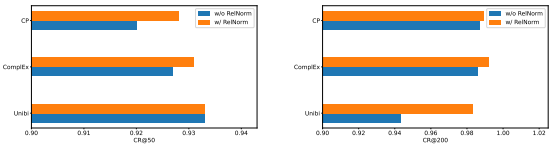
5 Analysis

In this section, we mainly discuss: 1) What are the key factors in PKGC, for which we have conducted a discussion related to the regularization term (**RQ1**); 2) The effectiveness of the acceleration module we proposed (**RQ2**); 3) Improving this task from the perspective of incremental learning,



(a) WN18. (b) FB15k.

Figure 4: Figures illustrating the dynamic completion process for various models on the WN18 and FB15k datasets. The closer the trend of the curve is to the upper left, the more efficient the model is in performing the dynamic completion. It is evident that UniBi-O(2) and UniBi-O(3) maintain a significant advantage on both datasets, while TransE performs poorly on both.



(a) (b)

Figure 5: Ablation studies of relation normalization (RelNorm) on (a) WN18 and (b) FB15k. We utilize CP and ComplEx as examples.

Module			Efficiency	
Root Filter	warm-up	SVF	Time	Speed-up
			$\approx 243d14h$	$1\times$
		✓	$\approx 32d4h$	$7.6\times$
✓			$2h10m52s$	$2{,}698.7\times$
✓		✓	$2h7m33s$	$2{,}763.7\times$
✓	✓		$3m38s$	$92{,}337.2\times$
✓	✓	✓	$50s$	$421{,}057.6\times$

Table 3: Ablation studies on proposed acceleration modules. The results of first two rows are estimated.

providing a feasible path for future research (**RQ3**); 4) Whether PKGC can execute in low-resource situation (**RQ4**).

5.1 RQ1: How Normalization Work in PKGC?

Given UniBi's remarkable performance over other models, we delve into an investigation to uncover the factors contributing to its strength. UniBi's unique feature lies in its normalization of both entities and relations, which is unusual among bilinear-based models. We argue that UniBi's performance enhancement primarily stems from its ability to make triples comparable. To illustrate, consider two triples: $f_1 = (h, r_1, t)$ and $f_2 = (h, r_2, t)$, with corresponding scores $s(f_1) = a$ and $s(f_2) = b$, where $a > b$. Without normalization, determining the relative plausibility of f_1 and f_2 is challenging due to the unaccounted score range. For instance, consider the case where $|\mathbf{e}| \leq 1$ and $s(\cdot) = \rho_r \cdot \mathbf{h}^\top \mathbf{t}$, confined within $[-\rho_r, \rho_r]$, with $\rho_{r_1} = 2a$ and $\rho_{r_2} = b$. Here, $s(f_2) = b$ achieves the highest score under relation r_2, while $s(f_1) < 2a$ still has potential for improvement. Thus, f_2 appears more plausible than f_1. UniBi distinguishes itself from other bilinear-based models by ensuring that $s(\cdot)$ falls within $[-1, 1]$ for any triple, enabling more rational comparisons between relations. We present CP and ComplEx as case studies, showing that both models improve their performance with relation normalization, as shown in Figure 5. We suggest that this advantage becomes more apparent when the number of triples per relation is limited, shedding light on why UniBi outperforms on FB15k but not on WN18.

5.2 RQ2: Ablation on Acceleration Modules w.r.t. Time cost

We commence by demonstrating the efficacy of two acceleration modules proposed for knowledge mining. Results in Table 3 reveal that all three modules contribute significantly to expediting the mining process, with the root filter exhibiting the most substantial speedup, exceeding 2,000 times.

	Origin		Retraining		Fine-tuning	
Model	MOAR	CR@50	MOAR	CR@50	MOAR	CR@50
CP	0.515	0.920	0.570	0.921	0.570	0.920
RotatE	**0.768**	**0.935**	**0.766**	0.935	0.753	0.928
ComplEx	0.630	0.927	0.631	0.927	0.637	0.928
QuatE	0.759	0.933	0.765	0.935	**0.764**	**0.933**
UniBi-O(2)	0.761	0.934	**0.766**	**0.936**	0.758	0.931

Table 4: Results of different strategies on WN18 dataset.

	WN18			FB15K		
Model	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$
CP	0.510	0.758	0.920	0.841	0.934	0.987
RotatE	0.560	**0.806**	**0.935**	0.850	0.946	0.987
RESCAL	0.521	0.795	0.892	0.821	0.932	0.988
UniBi-O(2)	**0.579**	0.797	0.934	0.894	0.963	0.994
UniBi-O(3)	0.578	0.799	0.934	**0.907**	**0.964**	**0.995**

Table 5: CR@k of UniBi-O(2) and CP on low-resource WN18 and FB15k. ($k = 50$ for WN18 and $k = 200$ for FB15k, MOAR results are in Appendix E).

More discussion are in Appendix D.

5.3 RQ3: Incremental Aspect on PKGC

In order to harness freshly validated knowledge, we employ the approach of incremental learning through retraining and fine-tuning, incorporating both existing and recently confirmed facts into the ongoing training process. We conduct incremental model training at a specified frequency denoted as Δs ($\Delta s = 5$ in our setting). During retraining, our training data encompasses $\mathcal{F}_{known}$, which comprises the verified facts from the most recent Δs steps denoted as $\mathcal{F}_{new}$. Conversely, during fine-tuning, our training is exclusively focused on the facts in $\mathcal{F}_{new}$ from the recent Δs steps. Additionally, we introduce an additional term to ensure that the previously acquired entity representations undergo moderate alterations (Formulated expression in Appendix B).

The outcomes, as depicted in Table 4, pertaining to the WN18 dataset, reveal that retraining often proves effective in bolstering performance, whereas fine-tuning may have adverse consequences. During progressive mining, at every stage, the recently acquired facts consistently receive high rankings. This implies that the model inherently possesses confidence in accurately identifying these factual triplets. Consequently, the impact of incorporating these new data into incremental learning is relatively modest. As emphasized in the context of active learning (Ren et al., 2022), it is crucial to emphasize samples near the decision boundary, particularly those involving low-ranking factual triplets.

5.4 RQ4: Low-resource PKGC

In this section, we explore low-resource PKGC to showcase its extensive practical applications. To achieve this, we reduce ρ, resulting in training the model on a smaller $\mathcal{K}_{known}$ while exploring the knowledge within a larger $\mathcal{F}_{un}$.Table 5 demonstrates that even in a low-resource scenario, UniBi ultimately attains reasonably satisfactory completion rates. On the WN18 dataset, when $\mathcal{K}_{known}$ holds only 30% of the knowledge, UniBi achieves nearly 60% completeness after 50 rounds of mining. With an initial knowledge base comprising 50%, this figure increases to almost 80%. On the FB15k dataset, after 200 completion rounds, UniBi enhances a knowledge base with a 50% completeness rate to nearly 90%. Similarly, concerning the CR@k metric, UniBi consistently retains its ranking position in comparison to the CP model.The results demonstrate that even in resource-constrained settings, PKGC consistently yields favorable outcomes, establishing it as a pragmatic experimental framework.

6 Conclusion

In this paper, we introduce a novel task called PKGC, which offers increased realism and a fundamental distinction by incorporating a finite verifier, and shifts the status of the model to that of a candidate knowledge proposer.In order to make the task more cost-effective, we have devised an optimized top-k algorithm that integrates with the semantic validity filter to significantly expedite the mining process in PKGC. Furthermore, We conduct a comprehensive series of experiments to demonstrate the performance of baseline models and to analyze factors that distinguish them. Our findings indicate that previous metrics and knowledge are insufficient for accurately predicting and explaining phenomena within PKGC. Additionally, We have preliminarily explored the effectiveness of some incremental learning methods within the context of PKGC, highlighting the potential for further development of the task.

Therefore, we conclude that PKGC not only serves as a novel benchmark but also represents a valuable avenue for investigating models' behavior in a more direct reflection of real-world scenarios. Future research directions include the pursuit of a more effective model and the exploration of methods to adaptively integrate incremental learning into the PKGC framework.

7 Limitations

This paper does not consider large KGs like ogbwiki2 (Hu et al., 2020). We recognize that experimentation on large datasets is crucial. However, our focus was to propose a new task that reduces the threshold of experimentation, enabling researchers to quickly iterate on training methods, models, and so on. We believe that experimentation on large datasets can be conducted later.This paper only considers two simple continual learning methods. More advanced and sophisticated methods for continual learning in PKGC should be explored.

References

Bo An, Bo Chen, Xianpei Han, and Le Sun. 2018. Accurate text-enhanced knowledge graph representation learning. In *NAACL-HLT*, pages 745–755.

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. Multi-relational poincaré graph embeddings. In *NeurIPS*, pages 4465–4475.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *ACL*, pages 6901–6914.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. Hitter: Hierarchical transformers for knowledge graph embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10395–10407. Association for Computational Linguistics.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818.

Luis Galárraga, Simon Razniewski, Antoine Amarilli, and Fabian M. Suchanek. 2017. Predicting completeness in knowledge bases. In *WSDM*, pages 375–383, Cambridge, UK.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *AAAI*, pages 4832–4839.

Frank L Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans Neural Netw Learn Syst.*, pages 1–21.

Jiayi Li, Ruilin Luo, Jiaqi Sun, Jing Xiao, and Yujiu Yang. 2023. Prior bilinear based models for knowledge graph completion. *CoRR*, abs/2309.13834.

Jiayi Li and Yujiu Yang. 2022. Star: Knowledge graph embedding by scaling, translation and rotation. *arXiv preprint arXiv:2202.07130*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*, pages 3243–3253.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Xiyang Liu, Huobin Tan, Qinghong Chen, and Guangyan Lin. 2021. RAGAT: relation aware graph attention network for knowledge graph completion. *IEEE Access*, 9:20840–20849.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *NAACL-HLT (2)*, pages 291–296.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT (2)*, pages 327–333.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, pages 809–816.

Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. 2021. An embedding-based approach to rule learning in knowledge graphs. *IEEE Trans. Knowl. Data Eng.*, 33(4):1348–1359.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela

Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023. Large language models and knowledge graphs: Opportunities and challenges. *CoRR*, abs/2308.06374.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. 2022. A survey of deep active learning. *ACM Comput. Surv.*, 54(9):180:1–180:40.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*, volume 10843, pages 593–607.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019a. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. 2019b. A re-evaluation of knowledge graph completion methods. *arXiv preprint arXiv:1911.03903*.

Zhaoxuan Tan, Zilong Chen, Shangbin Feng, Qingyue Zhang, Qinghua Zheng, Jundong Li, and Minnan Luo. 2023. KRACL: contrastive learning with graph context modeling for sparse knowledge graph completion. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pages 2548–2559. ACM.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, volume 48, pages 2071–2080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Rui Wang, Bicheng Li, Shengwei Hu, Wenqian Du, and Min Zhang. 2020. Knowledge graph embedding via graph attenuated attention networks. *IEEE Access*, 8:5212–5224.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *EMNLP*, pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*, pages 2659–2665.

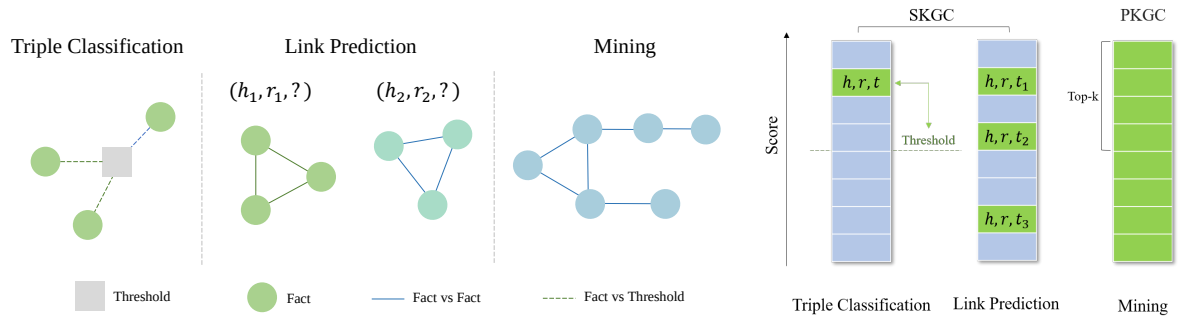
Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *NeurIPS*, pages 2731–2741.

Xuanyu Zhang, Qing Yang, and Dongliang Xu. 2022. Trans: Transition-based knowledge graph embedding with synthetic relation representation. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 1202–1208. Association for Computational Linguistics.



(a) Scope of connectivity.

(b) Scope of comparison

Figure 6: Two perspectives are used to explain the differences between PKGC and SKGC. Figure (a) provides clarification from the perspective of connectivity. Figure (b) presents it based on the scope of comparison. We use columns to denote all possible facts, and mark facts to be compared with green.

A Comparison between PKGC and SKGC

A.1 Connectivity Perspecitve

In this section, we delve into an examination of the task disparities with a specific focus on their connectivity attributes. This analysis involves treating each fact as a vertex and each comparison between facts as an edge, thereby evaluating the connectivity of the resultant graph. As elucidated in Figure 6a, our findings reveal the following distinctions: (a) triple classification fails to establish any form of graph due to its exclusive reliance on comparisons between facts and a predefined threshold, (b) link prediction yields outcomes for distinct connected components associated with different queries, and (c) mining leads to the creation of a connected graph.

It is essential to note that in cases where a Knowledge Graph regresses into a homogenous graph characterized by a single relation, there is inevitably only one connected component, which ensures the graph's connectedness. This observation reinforces our belief that, link prediction emerges as a more suitable task in the context of Graph Neural Networks (GNN) instead of KGE.

A.2 Comparison Perspective

Within this section, our objective is to elucidate the disparities inherent in the evaluation tasks of SKGC and PKGC. We focus on the scope of comparison, in addition to recognizing the apparent distinction in progressiveness.

SKGC encompasses two fundamental tasks: triple classification and link prediction. As depicted in Figure 6b, triple classification entails the comparison of each fact with a predefined threshold, while link prediction involves juxtaposing facts within the same query (h, r). However, what distinguishes these tasks is the absence of a mechanism for comparing facts between different queries. Consequently, in these tasks, facts remain isolated and segmented in terms of comparison.

Conversely, the mining process within PKGC mandates a global scope of comparison, permitting the evaluation of any two facts during the process of identifying the top-k facts. This expansive scope is not an inherent feature of SKGC tasks but rather arises as a necessity due to the involvement of realistic verifiers. Given the constraints imposed by limited verifiers[1], a mining process becomes a vital component, serving to deliver the most valuable candidates.

In light of these considerations, we contend that the scope represents another pivotal distinction between SKGC and PKGC, complementing the aspect of progressiveness. It is crucial to recognize that this distinction cannot be overcome merely by expanding the number of test samples in SKGC tasks. Even in scenarios where these tasks undergo exhaustive testing, encompassing all possible facts, their comparisons would continue to be confined to individual or partial assessments, failing to adopt a holistic perspective.

B Supplementary Details in Incremental Training

It is noteworthy that we do not apply a uniform regularization term for relation representations, as

[1] In cases where verifiers are limitless, they could exhaustively enumerate all possible facts, rendering the need for any model obsolete.

	WN18				FB15K			
Model	0.001	0.003	0.005	0.01	0.001	0.003	0.005	0.01
RotatE + F2	0.763	0.764	0.768	0.766	0.597	0.599	0.609	0.621
RotatE + DURA	0.759	0.760	0.760	0.763	0.616	0.614	0.607	0.571
QuatE + F2	0.757	0.759	0.757	0.757	0.637	0.628	0.605	0.583
QuatE + DURA	0.758	0.757	0.752	0.756	0.629	0.582	0.560	0.509

Table 6: Results of MOAR obtained from RotatE and QuatE in different regularization weights.

different models handle relations in diverse ways.

$$Reg_c(h,r,t) = ||\mathbf{h}_{new} - \mathbf{h}_{old}|| + ||\mathbf{t}_{new} - \mathbf{t}_{old}||, \tag{3}$$

In this context, $\mathbf{e}_{new}$ denotes the updated entity embedding, whereas $\mathbf{e}_{old}$ signifies the entity embedding prior to the update.We have detailed the exact loss equations for retraining and fine-tuning in Equations 4 and 5.

$$\mathcal{L}_{retrain} = -\sum_{(h,r,t)\in\mathcal{F}_{known}} \log\left(\frac{\exp(s(h,r,t))}{\sum_{t'\in\mathcal{E}}\exp(s(h,r,t'))}\right) + \lambda\cdot Reg(h,r,t) + \mu\cdot Reg_c(h,r,t) \tag{4}$$

$$\mathcal{L}_{new} = -\sum_{(h,r,t)\in\mathcal{F}_{new}} \log\left(\frac{\exp(s(h,r,t))}{\sum_{t'\in\mathcal{E}}\exp(s(h,r,t'))}\right) + \lambda\cdot Reg(h,r,t) + \mu\cdot Reg_c(h,r,t) \tag{5}$$

C Hyperparameters in Training

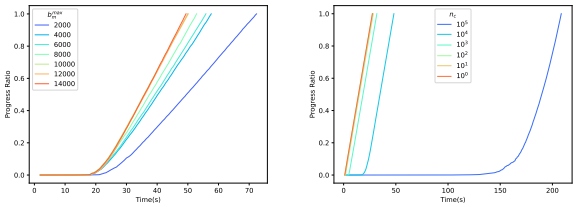
To fully demonstrate the model's potential, we systematically explore hyperparameters, as listed in Table 7, utilizing the verification data. Of particular significance are the hyperparameters related to regularization, where weights are chosen from a range of values, specifically 0.0, 0.001, 0.003, 0.005, 0.01. The ultimate selection of certain hyperparameters is documented in Table 8. Table 6 displays the MOAR results for the RotatE and QuatE models, achieved through the utilization of DURA and F2 regularization techniques.

Table 7: Details of hyperparameters.

Hyperparameters	Values
Batch size(Pretrain)	1000
Learning rate(Pretrain)	0.001
Learning rate(Retrain & Finetune)	0.001
Embedding dimension	500
Regularization weight	{0.0, 0.001, 0.003, 0.005, 0.01}
Update frequency(Retrain & Finetune) Δs	5
Max epoch(Pretrain)	100
Epoch(Retrain & Finetune)	20
Max batch size of Root Filter b_m^{max}	10000
Regularization Term	{F2, DURA}

Table 8: Optimal hyperparameter setting after search.

	WN18		FB15K	
Model	Regularization	Weight	Regularization	Weight
TransE	F2	0.003	F2	0.003
CP	F2	0.001	F2	0.0
RotatE	F2	0.005	F2	0.01
RotE	F2	0.01	F2	0.01
ComplEx	DURA	0.001	DURA	0.001
QuatE	F2	0.003	F2	0.001
RESCAL	F2	0.001	F2	0.003
UniBi-O(2)	DURA	0.01	DURA	0.01
UniBi-O(3)	DURA	0.01	DURA	0.01



(a) (b)

Figure 7: Ablation studies on (a) maximum mining batch size b_m^{max} and (b) number of candidate n_c in mining process.

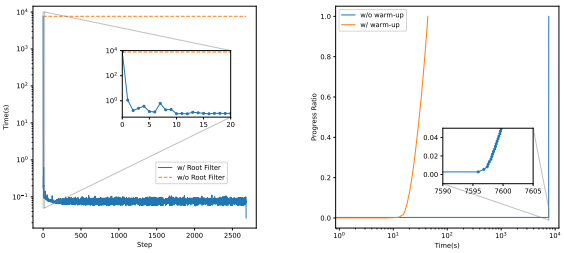
D Supplementary Details of Proposed Modules Ablation Study

D.1 Ablation on Hyperparameters in Mining

In this section, we examine the influence of hyperparameters on the mining process, with a particular focus on the maximum batch size for mining, denoted as b_m^{max}, and the number of candidates, represented as n_c. Our results, as depicted in Figure 7a, demonstrate that an increase in b_m^{max} leads to accelerated completion. Nevertheless, we also note that this acceleration effect saturates as the batch size reaches a specific threshold. We attribute this saturation to the interplay between CPU computations and communication with the GPU.

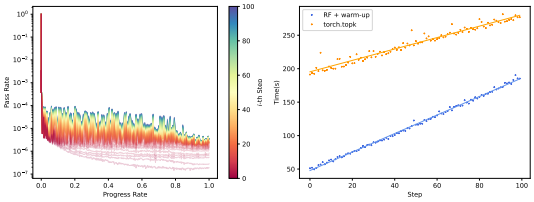
D.2 Detailed Ablation Studies w.r.t Time Cost

we observe that the root filter alone does not suffice to optimize the process. Furthermore, we have



(a) (b)

Figure 8: Ablation studies of (a) root filter and (b) warm-up modules. The dot lines is estimated.



(a) (b)

Figure 9: Ablation studies on pass rate and time consumption in different step. (a) Pass rate in different progresses and steps. The color represents the step, more small the step, more red the line, and more large the step, more blue the line. (b) Mining time during the steps. Here we also demonstrate that the combination of root filter (RF) and warm-up is faster than the implementation of topk on torch.

discovered that the SVF module fails to yield significant improvements in the absence of the warm-up module. This limitation is attributed to the fact that the naive root filter consumes the majority of time in the initial batch, as depicted in Figure 8a. To address this, Figure 8b illustrates how the warm-up module effectively mitigates congestion in the first batch, thus highlighting its importance.

D.3 Efficiency of Optimized Top-k

Due to the accumulation of visited facts suggested by Algorithm 1 line 6, the speed of the algorithm decreases with the number of steps. Specifically, as the number of steps increases, the amount of known knowledge increases correspondingly, while the heap decreases its update frequency. Consequently, the efficiency of our algorithm decreases with the number of steps.

As depicted in Figure 9a, the pass rate, representing the percentage of triples retained and sorted, exhibits two distinct trends. On one hand, the pass rate declines with the progression of mining in each step, influenced by the growing heap root that filters more triples. Conversely, the pass rate increases with the number of steps, as a substantial number of triples have been visited in prior steps, resulting in less frequent heap updates. Notably, the algorithm's effectiveness persists despite the declining pass rate.

Despite a decrease in the quantity of filtered triples with increasing steps, the algorithm's execution time demonstrates a linear growth, maintaining an acceptable range of performance, as depicted in Figure 9b. Furthermore, our algorithm exhibits superior performance compared to the naive and incomplete implementation reliant on $torch.topk$.

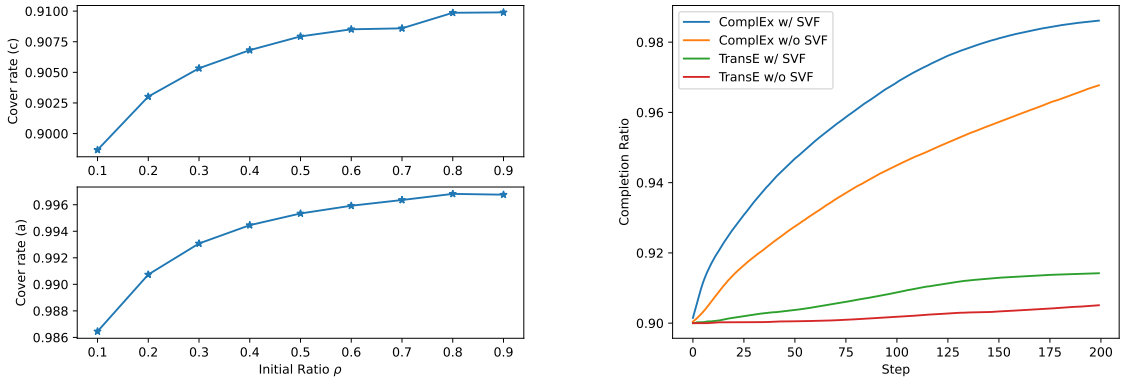
In summary, our experiments provide compelling evidence for the algorithm's effectiveness.

It is crucial to acknowledge that the observed variations with the number of steps are contingent upon the assumption that the number of candidates n_c is considerably larger than the size of the knowledge graphs. In the context of a larger knowledge graph with the same n_c, these variations would be less pronounced.

D.4 Performance after SVF

Initially, we establish a theoretical basis to affirm the feasibility of SVF. Figure 10a illustrates the coverage of SVF at various proportions represented by ρ. We examine two scenarios: the conservative case, where entities lacking class information are excluded from the coverage rate calculation, and the actual case, which includes these entities in the calculation. Both curves exhibit relatively minor variations across different values of ρ. In the actual case, the coverage rate is notably high, with an average of 0.99, a level of accuracy suitable for real-world applications. As an example, at $\rho = 0.8$, the data missed amounts to only 0.08%, a negligible fraction.

In preceding sections, we have established the effectiveness of SVF in terms of time efficiency. In this section, we assess SVF's effectiveness with regard to performance, focusing on the ComplEx and TransE models. Figure 10b demonstrates a significant enhancement in the models' performance resulting from the application of SVF. By using the ComplEx and TransE models for testing, we observe a significant improvement in both filtering efficiency and final completion rates following the integration of SVF. Clearly, without semantic fil-



(a) WN18 (b) FB15k

Figure 10: (a) The cover rate of SVF in different initial ratio ρ. Since there are entities missing the class information, we consider two extreme situations. up: c means conservative, which excludes all these entities. a means actually, which includes these entities and is the case in real situation. (b) SVF ablation on performance.

Table 9: MOAR of UniBi-O(2) and CP on low-resource WN18 and FB15k.

	WN18			**FB15K**		
Model	$\rho = 0.3$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.5$	$\rho = 0.7$	$\rho = 0.9$
CP	0.189	0.343	0.515	0.428	0.510	0.588
RotatE	0.307	0.364	0.768	0.513	0.610	0.621
RESCAL	0.237	0.458	0.545	0.419	0.533	0.668
UniBi-O(2)	0.296	0.428	0.761	0.603	0.711	0.841
UniBi-O(3)	0.275	0.451	0.762	0.630	0.720	0.846

tering, numerous intrinsically unreasonable triples disrupt the model's filtering operations during each mining round, leading to reduced efficiency.

E Supplementary Results of Low-resource PKGC

In addition to the CR@k results presented in Table 5, we show the results for the MOAR metric in Table 9.